ECS736P Assignment 2:
Spotify Podcast Search Engine Design

Group 84: Tom Allen, Tom Bomer, Chuks Obi and Tyler Bonnet

1: Overview (Problem Statement and Approach)

Booming demand has led to a growth in the supply of podcasts in recent years. There is now a wealth of podcast data that can be analysed in the field of Information Retrieval. In addition to the scale of global podcast data, there is also a wide variety of content, from current affairs to celebrity interviews and movie reviews. A search engine that can return relevant podcasts to the user will offer significant utility.

This project aims to create a search engine which returns relevant podcast episodes from the Spotify Podcast dataset (podcastsdataset.byspotify.com) based upon podcast user queries. These queries may represent a user wishing to find a known podcast, an episode which they previously listened to (but cannot remember the name of), or searching based upon a topic. We aim to use metadata to index podcast episodes using PyTerrier's indexer, which includes stop-word removal and stemming functionality. PyTerrier also provides an experiment function, which we intend to use to test the performance of a variety of models in retrieving relevant podcasts using metadata. Our search engine will then be built using the best-performing model based on mean average precision (MAP) and normalised discounted cumulative gain (NDCG). The finished product will allow users to search for podcasts in the dataset with their own queries.

Additional features

If there is enough time, we intend to use machine learning techniques with PyTerrier's Learning to Rank (LTR) function to see if the MAP and NDCG of our model can be improved: Examples of Retrieval Pipelines. With LTR, results can be re-ranked by implementing, for example, RandomForestRegressor from scikit-learn, LambdaMART from xgBoost, gradient boosted regression tree from LightGBM, coordinate ascent from FastRank, or Bidirectional Encoder Representations from Transformers (BERT)-based features.[1]

2: Dataset

We use the Spotify Podcast Dataset, which contains metadata, audio files and transcripts for over 100,000 English podcast episodes. The dataset was originally designed for the TREC conference Podcast Track, where the challenge was to identify relevant two minute sections of podcast from the dataset.

We set ourselves a different task, aiming to retrieve entire podcast episodes which contain relevant information. We plan to use metadata to identify relevant podcasts, treating each episode as a document. The metadata contains:

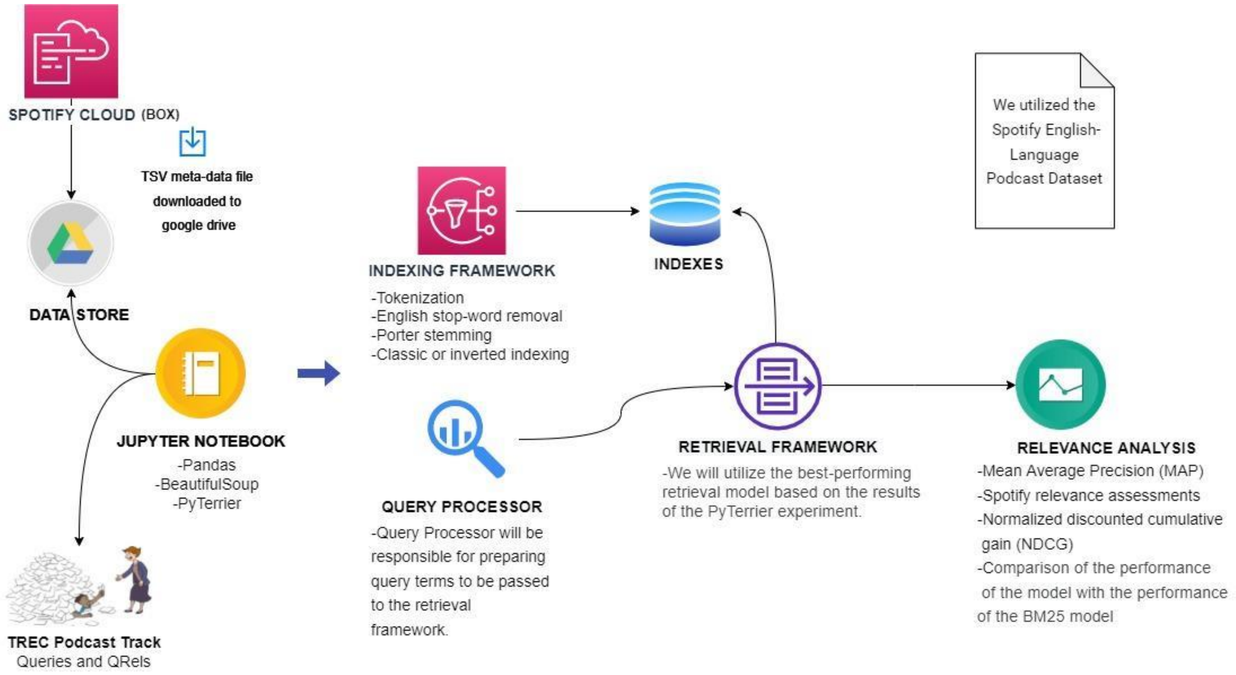
- Show URI
- Episode URI
- **Show name**
- **Episode name**
- **Show description**
- **Episode description**
- Publisher
- Language
- RSS link
- Duration

Specifically, we aim to use show/episode names and descriptions to identify relevant documents. Ground truth will be labelled from the TREC Podcasts Track 2020 training set. Relevance judgements score podcast sections on a scale of 0-4. We will use these relevance judgements as if they represent the relevance of the entire podcast episode.

[1] For examples of implementations of machine learning techniques using PyTerrier's LTR, please see: Wang, X. et al. (2020) *University of Glasgow Terrier Team at the TREC 2020 Deep Learning Track* In n *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC 2020)*. Available at: trec.nist.gov/pubs/trec29/papers/uogTr.DL.pdf. (Accessed 18 March 2022); github.com/terrier-org/pyterrier/blob/master/docs/ltr.rst; and colab.research.google.com/github/terrier-org/pyterrier/blob/master/examples/notebooks/ltr.ipynb#scrollTo=iGw58PCuumuT

3: Architecture

The diagram below shows the search engine software components. It illustrates each element starting from our datastore to our final evaluation metrics that will be developed or extended.


SPOTIFY CLOUD (BOX)
TSV meta-data file downloaded to google drive
DATA STORE
JUPYTER NOTEBOOK
-Pandas
-BeautifulSoup
-PyTerrier
TREC Podcast Track
Queries and QRels
INDEXING FRAMEWORK
-Tokenization
-English stop-word removal
-Porter stemming
-Classic or inverted indexing
INDEXES
QUERY PROCESSOR
-Query Processor will be responsible for preparing query terms to be passed to the retrieval framework.
RETRIEVAL FRAMEWORK
-We will utilize the best-performing retrieval model based on the results of the PyTerrier experiment.
We utilized the Spotify English-Language Podcast Dataset
RELEVANCE ANALYSIS
-Mean Average Precision (MAP)
-Spotify relevance assessments
-Normalized discounted cumulative gain (NDCG)
-Comparison of the performance of the model with the performance of the BM25 model

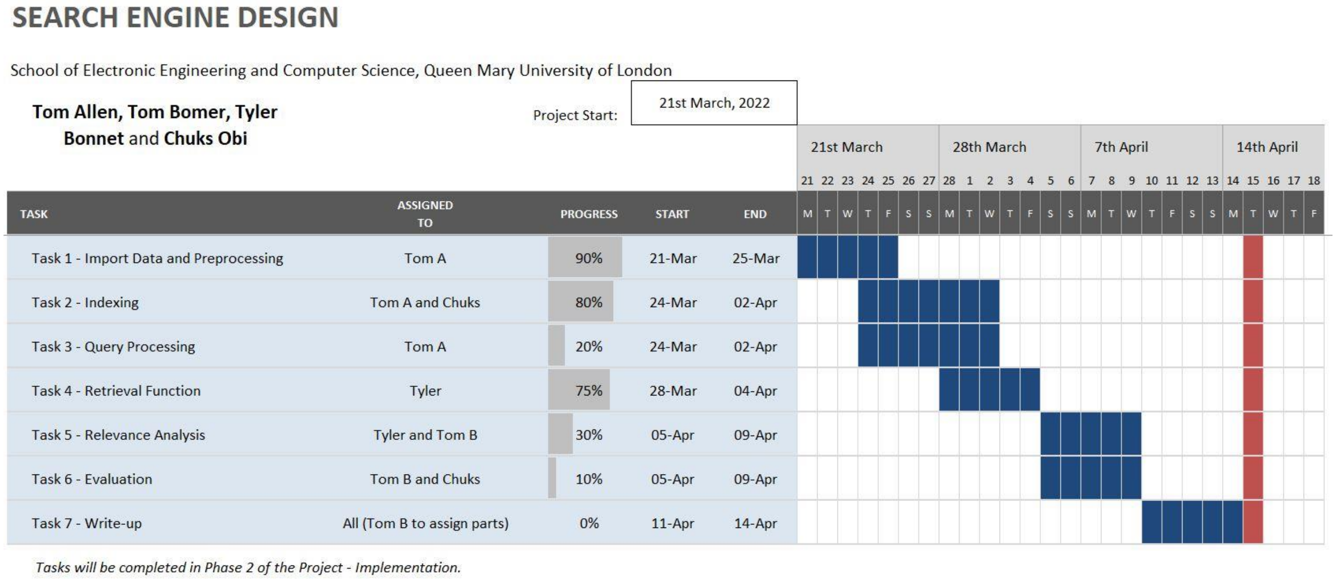
- ***Spotify Cloud:*** This is Box, a secure content management solution, it is the primary location of Spotify's English Language Podcast dataset.
- ***DataStore:*** Our data store is Google Drive, it is a cloud-based storage solution that allows you to save files online and access them anywhere. A .tsv metadata file was downloaded from the spotify cloud and was uploaded to google drive.
- ***Jupyter Notebook:*** The Jupyter Notebook is a web-based interactive computing platform. Our notebook is served by Google Colab. Here, data will be preprocessed and formatted
- ***Indexing Framework:*** PyTerrier is used to index podcast metadata. Tokenization is performed first, before English stop-word removal and Porter stemming. PyTerrier allows for classical indexing and single-pass, inverted indexing, so we will experiment with both.
- ***Retrieval Framework:*** Experiment and Model Selection

    - PyTerrier provides a library of retrieval models: Package org.terrier.matching.models. PyTerrier's experiment function allows us to easily run an experiment on the Spotify dataset using several models at once and measure performance based on various evaluation metrics: Running Experiments — PyTerrier. The best-performing model based on MAP and NDCG will be used for the search engine
- ***Evaluation Metrics***
    - MAP
        - MAP = (average precision for a given query) / (number of queries) = AveP(q) / Q
    - NDCG
        - NDCG = (actual DCG values) / (ideal DCG values)
    - MAP is the preferable metric when the models produce binary ratings - relevant or non-relevant. NDCG is preferable for non-binary models because it takes into account the graded relevance values.

4: Framework and Tools

- ***Google Colab*** and ***Google Drive*** will enable collaborative work on our Jupyter Notebook
- ***Pandas*** will be used for importing the podcast metadata and for formatting it as a dataframe.We may also use pandas to remove podcasts which do not have show/episode descriptions.
- ***BeautifulSoup*** will be used for web scraping queries and relevance judgements from the TREC conference webpages to import them as a pandas dataframe
- ***re*** (regular expressions) may be used to preprocess the data and remove any non-English characters
- ***PyTerrier*** will be used for indexing, running the experiment, ranking and evaluation

5: Timeline



SEARCH ENGINE DESIGN

School of Electronic Engineering and Computer Science, Queen Mary University of London

Tom Allen, Tom Bomer, Tyler Bonnet and **Chuks Obi**

Project Start: 21st March, 2022

TASK	ASSIGNED TO	PROGRESS	START	END
Task 1 - Import Data and Preprocessing	Tom A	90%	21-Mar	25-Mar
Task 2 - Indexing	Tom A and Chuks	80%	24-Mar	02-Apr
Task 3 - Query Processing	Tom A	20%	24-Mar	02-Apr
Task 4 - Retrieval Function	Tyler	75%	28-Mar	04-Apr
Task 5 - Relevance Analysis	Tyler and Tom B	30%	05-Apr	09-Apr
Task 6 - Evaluation	Tom B and Chuks	10%	05-Apr	09-Apr
Task 7 - Write-up	All (Tom B to assign parts)	0%	11-Apr	14-Apr

Tasks will be completed in Phase 2 of the Project - Implementation.

6: Roles and Responsibilities

Group Member	Responsibilities
Tom A	Import Data, Data Preprocessing, Indexing, Query Processing
Tom B	Project Oversight, Retrieval Function, Model Evaluation, Documentation for Report (including allocation of work for the write-up)
Chuks	Data Sourcing, Repository Setup, Indexing, Model Evaluation
Tyler	Exploring PyTerrier, Retrieval Function, Relevance Analysis, Learning to Rank

7: References

- Flowchart Maker: Flowchart Maker & Online Diagram Software
- Spotify Podcast Dataset: Spotify Podcast Dataset
    - Ann Clifton, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, Gareth Jones, Jussi Karlgren, Ben Carterette, and Rosie Jones. 2020. 100,000 Podcasts: A Spoken English Document Corpus. *COLING 2020.* Available at: https://www.aclweb.org/anthology/2020.coling-main.519/
- TREC Podcast Track: TREC Podcasts Track
    - Rosie Jones, Ben Carterette, Ann Clifton, Maria Eskevich, Gareth J. F. Jones, Jussi Karlgren, Aasish Pappu, Sravana Reddy, and Yongze Yu. 2020. TREC 2020 Podcasts Track Overview. *Proceedings from the 29th Text Retrieval Conference (TREC). NIST.* Available at: https://arxiv.org/pdf/2103.15953.pdf
- Box: Box
- Pandas: Pandas
- BeautifulSoup: BeautifulSoup Web Scraping
- re (regular expressions): Regular Expressions
- PyTerrier: PyTerrier's documentation
- Terrier retrieval models: Package org.terrier.matching.models
- Wang, X. et al. (2020) *University of Glasgow Terrier Team at the TREC 2020 Deep Learning Track* In *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC 2020)*. Available at: trec.nist.gov/pubs/trec29/papers/uogTr.DL.pdf. (Accessed 18 March 2022).